COMP-202A, Fall 2007, All Sections
Assignment 4

Due: Tuesday November 13, 2007 (23:55)

You **MUST** do this assignment individually and, unless otherwise specified, you **MUST** follow all the general instructions and regulations for assignments.

Note that for this and all further assignments, graders **will generously** deduct marks for assignments that do not follow instructions and regulations.

Failure to respect instructions and regulations:	0 to -10 points
Part 1:	0 points
Part 2, Question 1:	3 points
Part 2, Question 2:	7 points
	10 points total

Part 1 (0 points)

Do not hand in this part. It will not be graded. But doing this exercise might help you to do the second part of the assignment (that will be graded). If you have difficulties with the questions of Part 2, then we suggest you go to one of the office hours. The TA can help you and work with you through the warm-up questions.

Warm-up Question 1 (0 points)

A Tribute to Paul Erdös: As a tribute to Paul Erdös' numerous collaborations with other mathematicians, the Erdös number has been created:

- Paul Erdös has number 0
- His direct collaborators have number 1
- Individuals who have written a paper not with Erdös but with someone whose Erdös number is 1 have Erdös number 2 and so on

Create a class ErdösNumber that keeps track of a person's name and his Erdös number. Your class should also have methods that return the name and the Erdös number of the individual and include a method for updating his Erdös number (note, that one's Erdös number never increases and can never become 0 nor 1). Once a researcher collaborates with a person of Erdös number s, his number is updated to $s + 1$. In your main method,

create a few (say 8) instances of this class using information from the user. Finally, display the name of the individual with the lowest Erdös number.

Part 2 (3 + 7 = 10 points)

The questions in this part of the assignment will be graded.

Question 1 (3 points)

A fly, two trains and a Gedankenexperiment! John von Neumann[1] was asked the following:

> Two trains 150 miles apart are traveling toward each other along the same track. The trains travel at 60 and 90 miles per hour respectively. A fly buzzes from the first train to the second, turns around immediately, flies back to the first train, and turns around again. It goes on flying back and forth between the two trains until they collide. If the fly's (constant) speed is 120 miles per hour, how far will it travel?[2]

He is famously known to have summed the infinite series correctly in a few seconds in his head. However, the correct answer can be calculated using a much simpler method by focusing on the trains and arguing that they are approaching each other in $60+90 = 150$ miles per hour, and since they're initially 150 miles apart, it 'll take them an hour to meet. The fly spends the same amount of time traveling and so it covers a total of 120 miles.

Your task is to create a class Train which models a train as an object with a constant speed and a starting position (relative to the origin). Ask the user to enter the speeds and locations of two trains, speed of the fly and create two trains based on the input and calculate the total distance travelled by the fly using the above simple calculation (and not the infinite series). You need to complete the implementation of the following methods in Train.java and TrainTest.java:

- A constructor
- Accessor methods
- A method that computes the initial gap between the two trains
- A method that calculates the distance the fly has flown using the following equation:

  $$\text{distance} = \text{fly's speed} \times \text{total time}$$

  $$\text{where total time} = \frac{\text{gap between the two trains}}{(\text{speed of the first train} + \text{speed of the second train})}$$

- A main method in which two trains are created based on user's input values, the fly's (constant) speed is set and the above calculation is performed and the result is displayed to the user.

[1] http://en.wikipedia.org/wiki/John_von_neumann

[2] http://mathforum.org/dr.math/faq/faq.fly.trains.html

Question 2 (7 points)

Secrets! Five top spies gather to learn an ultra-classified secret from their master spy. To sustain peace, the master spy encodes the classified secret (*a.k.a* CS) and gives each spy a portion of the encoded message so that the message can only be decoded once all five spies are present again and the secret message would forever be lost otherwise.

In this question, you 'll complete the implementation of a Spy class which also enables the spies to encode and decode messages and ultimately decode CS.

In Spy.java, you will declare how a Spy object is defined, messages are encoded and then decoded. In SpyTest.class, you will create five Spy objects (corresponding to the five top spies) and attempt to decode the Master Spy's CS.

Complete the implementations of Spy.java nad SpyTest.java using the following guidelines:

- **Guidelines:**
  - Each spy has a *name*, *a serial ID*, and a *password*:
    - A spy's *name* is a string of letters that constitute his first and last name
    - His *serial ID* is a 6-digit number that starts with 202, the next two digits are randomly generated and the last digit is his rank (his order of creation relative to the other spies). For instance, the first Spy object that is created will have rank 1 and his serial ID will end in 1.
    - His *password* is only known to him and is used to authenticate him. In the constructor, only an encoding of this password is stored.
  - All encodings and decodings use the the following permutation cipher:
    - **Encoding:** Divide the string into substrings of length 6, and move the character at index
      - 0 to character at position 1
      - 1 to character at position 3
      - 2 to character at position 5
      - 3 to character at position 4
      - 4 to character at position 0
      - 5 to character at position 2
    - **Decoding:** Reverse the above encoding effect by permuting the encoded string again (only in the reverse order)

- 0 to character at position 4
- 1 to character at position 0
- 2 to character at position 5
- 3 to character at position 1
- 4 to character at position 3
- 5 to character at position 2

* For instance,

  Encoding of "ABCDEF" is "EAFBDC"

  Decoding of "DCAAAN" is "CANADA"